Project #1 Command Line Use

Client

`tsapp -c` *`server`* `[options]` *`port`*

`-c`: run as client

server: Address of origin server to communicate with

port: destination port for communicating with server.

OPTIONS

`-u`: use UDP to communicate with server. *Client will default to UDP if neither –u nor –t is specified.*

`-t`: use TCP to communicate with server. *Client will default to UDP if neither –u nor –t is specified.*

`-z`: use UTC time to display time to user. Otherwise, use calendar format.

`-T <time>`: set server time (UTC). *Requires use of --user and --pass.*

`--user <name>`: credentials to use. *client & server applications*

`--pass <password>`: credentials to use. *client & server applications.*

`-n <#>`: number of consecutive times to query the server, otherwise a single query is performed.

Server

`tsapp -s -T <time> [options]` *`UDP_Port TCP_Port`*

`-s`: run as server

`-T <time>`: initial value of server time. Value is interpreted as UTC.

UDP_Port: UDP listening port to service client connections

TCP_Port: TCP listening port to service client connections

OPTIONS:

`--user <name>`: credentials required to modify (set) server time via client call. *Omitting the credentials implies no client is authorized to modify the server time.*

`--pass <password>`: credentials required to modify (set) server time via client call. *Omitting the credentials implies no client is authorized to modify the server time.*

Proxy Server

`tsapp -p` *`server`* `[options]` *`UDP_Port TCP_Port`*

`-p`: run as proxy

server: Address of origin server to communicate with

UDP_Port: UDP listening port to service client connections

TCP_Port: TCP listening port to service client connections

OPTIONS:

--proxy-udp <#>: UDP port to communicate with origin server. *Required if –t is not specified.*

--proxy-tcp <#>: TCP port to communicate with origin server. *Required if –u is not specified.*

`-u`: use UDP to communicate with server, regardless of client transport protocol. *If neither –u nor –t is specified, proxy will communicate with server based on client protocol.*

`-t`: use TCP to communicate with server, regardless of client transport protocol. *If neither –u nor –t is specified, proxy will communicate with server based on client protocol.*

Example Use Cases

Origin server at 127.0.0.5, single UDP client query

```
1. tsapp –s –T 5 5000 5001
2. tsapp –c 127.0.0.5 5000
```

Origin server at 127.0.0.5, single client TCP modify server time

```
1. tsapp –s –T 5 --user usr --pass pw 5000 5001
2. tsapp –c 127.0.0.5 –T 99 --user usr --pass pw -t 5001
```

Origin server at 127.0.0.5, pass through Proxy server at 127.0.0.4, UDP client query loop

```
1. tsapp –s –T 5 --user usr --pass pw 5000 5001
2. tsapp –p 127.0.0.5 --proxy-udp 5000 --proxy-tcp 5001 4000 4001
3. tsapp –c 127.0.0.4 –n 10 –u 4000
```

Origin server at 127.0.0.5, TCP only Proxy server at 127.0.0.4, UDP client query; user wants UTC

```
1. tsapp –s –T 5 --user usr --pass pw 5000 5001
2. tsapp –p 127.0.0.5 –t --proxy-tcp 5001 4000 4001
3. tsapp –c 127.0.0.4 –z –u 4000
```

Hints on Parsing Command Line Arguments

The GNU C Reference Manual is a reference for the C Programming Language as implemented by the GNU C compiler. The GNU Compiler collection (gcc) is widely used and well documented for use in project assignments. The Reference Manual provides examples for many functions, including argument parsing.

http://www.gnu.org/software/libc/manual/html_node/Program-Arguments.html#Program-Arguments

There may be various Java ports of GNU C library functions which can be used when appropriately cited in the project documentation and code as required by RIT's Academic Integrity standard as well as any licensing requirements of the third party library.

Real Life Example

`iperf` is a well-known command line program used to test network performance. Examining the man pages (help documentation) for this program may help demonstrate the flexibility of this style of user interface. (As a network application, `iperf` also defines its own protocol to communicate and calculate performance statistics between client and server!)

Example Output Formatting for `tsapp`

The following are provided as examples of previous output formatting. The output should be readable and communicate the information required by the project. You may choose to implement a –debug mode that provides more details for use in troubleshooting the application.

```
tsapp -c localhost 5000
Starting as Client
Using UDP
Querying server for the time
Sent on Tue, 08 Mar 2016 14:15:38 EST
Original server: 129.21.22.196 at 32 ms
Roundtrip time: 61 ms
Server time: 5
```

```
tsapp -s -T 5 --user usr --pass pw 5000 5001
Starting as Server
Setting server time to 5
Server started at: 5
Listening for UDP on Port: 5000
Listening for TCP on Port: 5001
User name: usr
Password: pw
Connected to 127.0.0.1
Client at 127.0.0.1 set server time to 99
Connection with 127.0.0.1 closed
Received UDP packet from /127.0.0.1
Connected to 127.0.0.1
…
```

```
tsapp -c localhost 5000

************************************

CLIENT information =>

Client trying to connect to server :: localhost

port number to connect to 5000

Client is getting ready for UDP connection to server. Please wait ...

UDP connection successful!!!

************************************

Requesting Time from Server.Please wait ...

================================================

CLIENT =======> GET_TIME_REQUEST =========> SERVER

GET_TIME_REQUEST : CLIENT TO SERVER

MESSAGE FORTMAT::

ATTRIBUTE_FIELD               ATTRIBUTE_LENGTH      ATTRIBUTE_VALUE

type                  2                     11

================================================

================================================

SERVER =======> GET_TIME_RESPONSE =========> CLIENT

MESSAGE FORTMAT::

ATTRIBUTE_FIELD               ATTRIBUTE_LENGTH      ATTRIBUTE_VALUE

type                  2                     12

code                  3                     202

time                  10                    5

================================================

Time received successfully!!!

Epoch Time at server : 5

Hops information between client and server :

IP_ADDRESS    ::   PORT_NUMBER   ::   RTT_TIME(ms)

129.21.22.196   ::   5000          ::   23

Total Hops present : 1

GoodBye!!!
```